MEGAnno: Exploratory Labeling for NLP in Computational Notebooks

Dan Zhang*, **Hannah Kim***, **Rafael Li Chen**, **Eser Kandogan**, **Estevam Hruschka**
Megagon Labs
{dan_z,hannah,rafael,eser,estevam}@megagon.ai

Abstract

We present MEGAnno, a novel exploratory annotation framework designed for NLP researchers and practitioners. Unlike existing labeling tools that focus on data labeling only, our framework aims to support a broader, iterative ML workflow including data exploration and model development. With MEGAnno's API, users can programmatically explore the data through sophisticated search and automated suggestion functions and incrementally update labeling schema as their project evolve. Combined with our widget, the users can interactively sort, filter, and assign labels to multiple items simultaneously in the same notebook where the rest of the NLP project resides. We demonstrate MEGAnno's flexible, exploratory, efficient, and seamless labeling experience through a sentiment analysis use case.

1 Introduction

Data labeling is an important step in the machine learning life cycle since the quality and quantity of training data directly affect the model performance (Geiger et al., 2021). Unfortunately, existing annotation tools tend to consider the data labeling step in isolation from the broader ML life cycle, ignoring the iterative workflow of researchers and practitioners. However, activities such as data selection, exploratory data analysis, data labeling, model training, and evaluation do not happen sequentially (Rahman and Kandogan, 2022). Instead, continuous iterations are required to improve data, annotation, and models (Hohman et al., 2020).

To further investigate this gap, we examine the data annotation practices within the ML model development life cycle. Based on a formative study with six researchers from our organization, we characterize their annotation practices as a "dual-loop" model shown in Fig. 1. After data preprocessing (Fig. 1 ①), researchers define their annotation schema in terms of what labels to collect, how many data points are needed, and so on (Fig. 1 ②). As they explore and annotate the data (Fig. 1 ③), they often go back and refine the annotation schema due to their improved understanding of the data and updated mental models for the tasks (Fig. 1 ④). For example, in a document classification task, a user may start with loosely defined category labels and add more choices as she discovers relevant documents (Felix et al., 2018). Throughout this paper, we refer to this cycle as **data understanding/exploration loop**. Next, the labeled data is exported from the annotation tool and is used to train a model (Fig. 1 ⑤). However, in practice, ML model training is rarely completed in one pass and usually goes through many iterations of labeling, training, data, and model debugging (Pustejovsky and Stubbs, 2012). Fig. 1 ⑥ refers to the cases where researchers may need to collect more data (e.g., for the less represented classes due to a suboptimal prediction performance of the downstream model) or further refine the schema. We call this cycle (i.e., model training, evaluating and debugging, collecting more data, and training again) **model evaluation loop**.

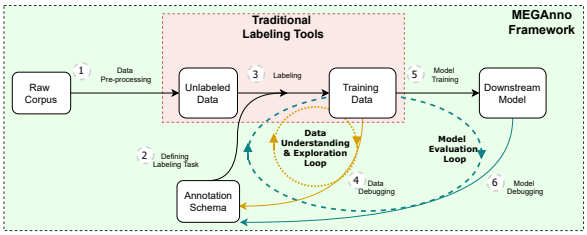

Figure 1: Dual-loop model for data annotation: (1) data understanding/exploration loop (yellow): iteratively update annotation schema while exploring and annotating data and (2) model evaluation loop (green): train and improve a downstream model over iterations by debugging data. Most tools are focused on the labeling step only (red box). MEGAnno aims to capture both loops seamlessly within the framework (green box).

We find that the iterative dual-loop workflow

* Equal contribution.

Proceedings of the Fourth Workshop on Data Science with Human-in-the-Loop (Language Advances), pages 1 - 7

of NLP researchers and practitioners is rarely supported in most existing tools. More specifically, we identified three main challenges towards supporting the full annotation life cycle:

- **Gaps between ML toolings.** Most of the existing tools are standalone and designed for a specific ML step, which results in frequent context switching and data migration overhead in the researchers' daily workflow.
- **Lack of customizable and granular control.** Not all data points are equally important. There are often cases where users might want to prioritize a particular batch (e.g., to achieve better class or domain coverage or focus on the data points that the downstream model cannot predict well). Although some recent active learning based tools (Montani and Honnibal, 2018; hua) can provide suggestions for the next batch, most tools do not offer customizable and fine-grained control with or with a downstream model (i.e., covering both loops).
- **Lack of support for project evolution.** Current annotation tools usually work with the assumption that the data collection task is well-defined and immutable and ignore that annotation projects can evolve as explorations happen and insights are gathered. Thus they lack the support to help users make evolution decisions, and their immutable nature makes it hard to apply these changes.

To address the challenges, we present MEGAnno, a flexible, exploratory, efficient, and seamless labeling framework for NLP researchers and practitioners. It provides a seamless experience where data pre-processing, annotation, analysis, model development and evaluation can happen in the same notebook, a popular daily working environment for data science practitioners. MEGAnno provides customizable interfaces to help users drive their project to the desired directions through rich heuristic-based search, automatic suggestion, and active learning based suggestions of the next data batch. With project evolution in mind, MEGAnno is designed to work with flexible task schema and provides a built-in analysis dashboard to aid decision-making. To our knowledge, MEGAnno is the first flexible, exploratory labeling framework that can support ML workflow seamlessly in computational notebooks (Fig. 1: green box).

2 MEGAnno

2.1 Framework Overview

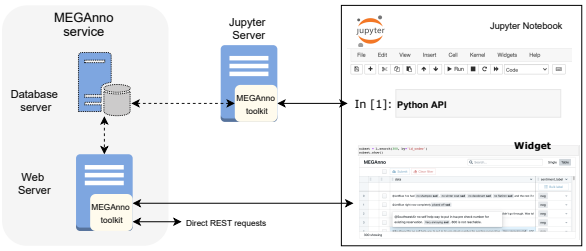

Figure 2: The MEGAnno framework provides exploratory annotation services through a toolkit (installable as Python libraries) providing programmatic interfaces, a web server providing language-agnostic REST APIs, and an internal database to store data, annotation, and related artifacts. Solid lines show programmatic interactions through Python APIs calls and REST calls delegated by our notebook widget or directly issued by authenticated applications. Dotted lines show internal interfaces where MEGAnno toolkit handles communication with the database and are hidden from the users.

MEGAnno provides service through 1) an internal database that stores the data, annotations, and various artifacts produced in the annotation process, 2) a MEGAnno toolkit that provides python API for programmatic and visual data exploration and labeling, and 3) a web server that provides language-agnostic REST APIs (Fig 2). After installing the toolkit on a Jupyter server, users will have access to our Python APIs and React-based widget to manage their project, explore and annotate their data from any connected notebook.

Data model A `Data record` refers to an item in the pre-processed data corpus for labeling. It can be a sentence, a paragraph, a document, or a flattened text from multiple texts such as a question-answer pair. A `Label` is the smallest unit of user labeling output. MEGAnno currently supports record-level (e.g., topics for document or sentence) and span-level (e.g., named entities) labels. An `Annotation` is a set of labels given to a data record by an annotator. `Metadata` refers to additional information related to the content of a data record (e.g., externally generated part-of-speech tags, embeddings) or of an annotation (e.g., time taken to label, disagreement among annotators). Such information can be helpful in various steps of the ML iterations. A `Subset` is a slice of the data records in the database. Subsets can be of random data records or can be generated through search queries that match certain characteristics.

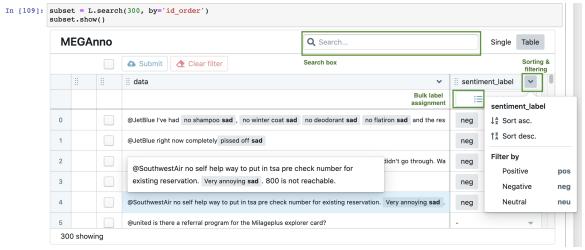

Figure 3: The table view to show multiple data records. Hovering over a data record shows its full text in a pop-up. This view allows exploration by searching, sorting, and filtering over labels and single/bulk annotation.

Task schema We support a wide variety of tasks through our customizable schema in JSON format. To collect a label, users need to specify the level (i.e., record or span) and provide a list of options to choose from. For a task, MEGAnno supports arbitrary numbers of both types of labels. We'll see concrete schema examples for a sentiment prediction and extraction task in Section 3. At any stage, users can always update the schema to reflect the evolution of the project. There are certain constraints to schema updates to keep the consistency of data. Adding new labels or new label options will always be allowed. Removal of labels and options will trigger a database query and MEGAnno will warn the user if there exist such labeled instances. [1]

2.2 MEGAnno Jupyter Notebook Widget

MEGAnno's interactive notebook widget features 1) our novel table view to facilitate exploratory and batch labeling and 2) the single view, which is similar to traditional labeling UIs.

Table view The table view (Fig. 3) shows data items in a `Subset` and their annotations if any. Each record-level label is shown as a column, and span-level labels are shown together with the highlighted textual span in the data column. Users can hover over an item to see full text in a pop-up. The search box supports three types of search (exact, fuzzy, and regex-based) to filter the data subset further. Users can sort and filter rows based on any record-level labels using the dropdown menu. To assign a record-level label, the users can click on the cell's arrow button and select from the dropdown options. Alternatively, the users can assign the same label to multiple records simultaneously by selecting those records and clicking the bulk label button.

Single view The table view is good for exploration, but the limited space makes span-level annotation cumbersome. So we also provide a single view where users can have a more zoomed-in experience. By clicking the "Single" button on the top-right corner or double-clicking on a data record, the widget switches to the single view (Fig. 4). In this view, users can assign record-level labels on the right side and span-level labels on the left side by selecting/dragging target spans and choosing the label from the options drop-down. Users can loop through the subset using the prev/next button based on the order specified in the table view. At any time, users can switch between the two views by clicking the top right buttons, and the widget preserves all uncommitted annotations during view changes.

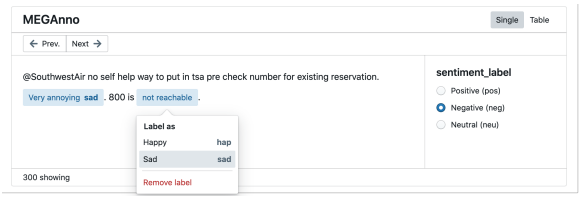

Figure 4: The single view to annotate data one by one. In this view, users can drag and label spans for extraction tasks.

Working with multiple annotators Annotation is rarely done by a single person. As an initial step towards collaborative annotation, MEGAnno provides virtually separated namespaces for each annotator. Users identify themselves by a unique authentication token while connecting to the service and only update their own labels through the widgets. MEGAnno provides a reconciliation view (Fig. 5) to look at labels from different individuals and resolve potential conflicts.

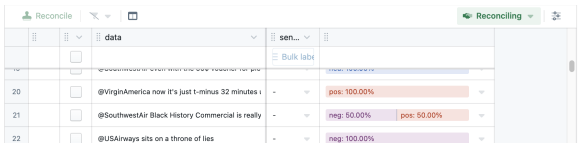

Figure 5: Reconcilation view showing the existing label distribution for data points.

Dashboard MEGAnno also provides a built-in visual monitoring dashboard (Fig. 6). As projects evolve, users would need to understand

[1] MEGAnno provides an option to clean up legacy labels and retry automatically.

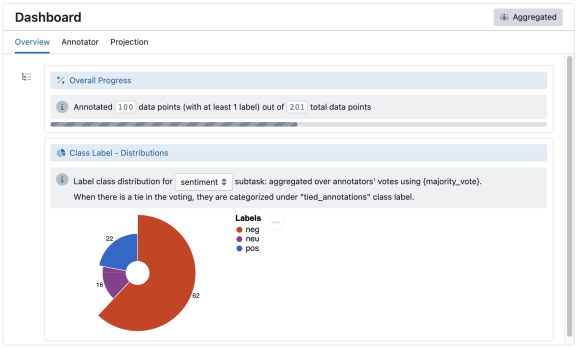

Figure 6: Dashboard widget to monitor the progress and statistics of the project and aid decision-making.

the project's status to make decisions about the next steps, like collecting more data points with certain characteristics or adding a new class to the task definition. To aid such analysis, the dashboard widget packs common statistics and analytical visualizations based on a survey of our pilot users. The "overview" panel shows statistics about overall progress and per-label class distribution. If multiple annotators are involved, the distribution reflects the majority vote over annotators [2]. The remaining "annotator" and "projection" panels are hidden due to space limitations. To help identify problematic annotators, the annotator pannel shows statistics like overall individual contribution and disagreement scores with others. The projection panel provides customizable visualization to project data points to a two-dimensional visual space. By default, we show the t-SNE (Van der Maaten and Hinton, 2008) projection of sentence bert (Reimers and Gurevych, 2019) embeddings.

2.3 MEGAnno APIs

Project management The management module provides various interfaces to configure and monitor the annotation project through the `Project` class. `import_data` loads the data records from CSV or JSON files into the database. `set_config` updates the project configuration as it evolves. `set_meta` assigns metadata (e.g.,POS tags, document embeddings) for each data record through user-defined functions. `get_status` returns the status of the project such as the number of annotated data records and detailed statistic about each label.

A critical feature of MEGAnno is to select interesting subsets of data to show in the widget. Subsets can be generated in a user-initiative way via our search engine or a data-driven way via automated suggestions.

Search for subsets MEGAnno supports sophisticated searches over data records, annotation, and user-defined metadata through the `Project.search` API. Users can search data records by keywords (e.g., documents mentioning "customer service") or regular expressions to express more complex patterns. The users can also search the database based on already assigned labels (e.g., records with a positive sentiment label). As will be explained later, MEGAnno acknowledges the value of auxiliary information for ML projects and provides advanced search functionalities over metadata. For example, users can query with patterns combining regex expressions and POS tags like `project.search("(best|amazing) <ADJ> <NOUN>", by="POS")`.

Automated subset suggestion Searches initiated by users can help users explore the dataset in a controlled way, but the quality of searches is only as good as users' knowledge or heuristic about the data and domain. MEGAnno provides an automated subset suggestion engine to assist the exploration. Users can customize the engine by plugging in external suggestion models as needed. Currently, the engine provides two types of techniques:

- **Embedding-based suggestions** makes suggestions based on data embedding vectors provided by the user. `Subset.suggest_similar` suggests neighbors of data in the querying subset. `Project.suggest_coverage` examines all the data records within the embedding space in an unsupervised way and suggests data points from the less annotated regions to improve annotation coverage of the corpus.

- **Active suggestions** utilizes active learning techniques to recommend the most informative data for the downstream model. With libraries like ModAL (Danka and Horvath), users can select from various selection strategies based on model uncertainty and entropy, etc. Since MEGAnno's seamless notebook experience covers the whole loop from annotation to model training and debugging, users

[2] Users can provide their aggregation function to resovle conflicts between annotators

can actively select a subset, annotate the subset, update the model, and test again in the same notebook without switching environments.

The output of selection engines are instances of the `Subset` class with the following methods: `show` returns a notebook widget for interactive exploration and annotation. `batch_annotation` sets the same record-level labels for all data records in the subset. `suggest_similar` returns a new subset of the database containing the most similar data for each record in the querying subset according to some metadata with valid distance functions.

3 Use Case: Sentiment Analysis

We present a use case to illustrate how MEGAnno can support data annotation in NLP researchers and practitioners' workflow. Meggie is a data scientist who wants to train a sentiment-related model for her project. She obtains a Twitter dataset[3] about US airlines and decides to label it using MEGAnno.

Import data In an empty notebook, Meggie starts by initializing the project named "Tweet Sentiment" using a MEGAnno Python API. She gets a copy of the data from the product manager who often uses Google spreadsheet and imports the data from its published link.

```
from meganno import Project
L = Project(<auth>, "Tweet Sentiment")
L.import_data(<doc_url>, format="csv")
```

Set up initial schema Without knowing much about the data, Meggie decides to start the project by collecting binary labels and setting up the project's schema. Knowing that MEGAnno supports flexible and editable schema, Meggie does not worry about getting the perfect schema in the first round and can start exploring and annotating.

```
label_schema = [{
    "label_name": "sentiment_label",
    "level": "record",
    "options": ["Positive", "Negative"]
}]
L.set_config(config1)
```

Explore and annotate She starts by exploring the first 300 data points in the widget's table view.

[3]The Kaggle dataset (for Everyone library) has ground-truth sentiment labels available. But for demonstration purposes, we ignore them and assume Meggie only gets the raw Tweets. The dataset contains 14K tweets about major US airlines scraped in February 2015.

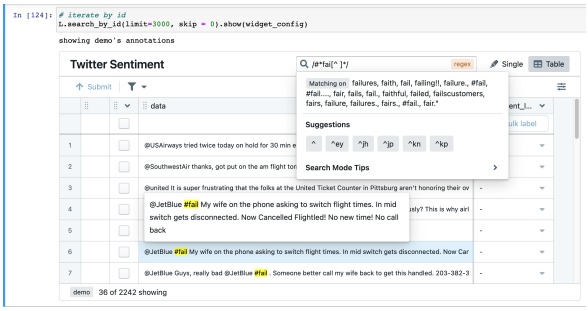

Figure 7: UI Search by regular expression. Matched keywords are highlighted.

Using the search box, she filters the subset with the keyword "amazing". As expected, most of the data records reflect a positive sentiment, so she assigns a positive label to multiple data items using the "Bulk label" button. Next, she wants to examine tweets with hashtags related to failing, so she tries regular expression search using `#fai[^ ]*` (Fig. 7). With better understanding of the dataset, Meggie chooses to perform more advanced exploration using part-of-speech metadata. She imports a POS tagger from spaCy (Honnibal and Montani, 2017) and retrieves tweets that match interesting patterns such as `best <NOUN> of <NOUN>`.

```
import spacy
tagger = spacy.load("en_core_web_sm")

pos1 = L.search("(best|amazing) <ADJ> <
    NOUN>", by="POS", tagger=tagger)
pos2 = L.search("best <NOUN> of <NOUN>",
     by="POS", tagger=tagger)
meganno.union(pos1, pos2).show()
```

With such an exploratory approach and the batch labeling feature, she can annotate much faster and in a more controlled way.

Candidate suggestion After a few more heuristics, she runs out of ideas, so she takes advantage of the suggestion feature. She selects the sentence bert (Reimers and Gurevych, 2019) encoder as the meta-data generation function. She first issues a query with strategy `similarity` to collect data points similar to the ones retrieved from the previous POS search. Finally, she wants to see samples from less covered areas in the embedding space to improve the diversity of the training data and issues a `coverage` query.

```
m = SentenceTransformer("all-MiniLM-L6")
# set metadata generation function
L.set_meta("bert", lambda x: list(m.
    encode(x)))
# get more data like the query result in
    the previous query(subset_pos)
```

```
subset_sim = pos2.suggest_similar(
    meta_name="bert")
# get data from less covered areas in
    the embedding space.
subset_cov = L.suggest_coverage(
    meta_name="bert")
```

Update the schema Meggie is now happy with the collected labels, but she aims to go one step further to understand which words or phrases lead to the sentiment judgment. So she updates her schema by adding a new span-level label called `sentiment_span` with label options being "happy" or "sad". Each data record can have arbitrary numbers of such span-level labels.

```
label_schema = [{
    "label_name": "sentiment_span",
    "level": "span",
    "options": ["Happy", "Sad"]
}, ... {# previous label options
}]
```

After updating the schema, she fetches all records with neutral labels in a widget. To highlight and annotate spans, she goes into the single view (as shown in Fig. 4). At any step of the iteration, she could refer to the dashboard widget to monitor the project progress. After several rounds of similar iterations, she feels good and concludes her exploration. Finally, Meggie can export the annotated data in JSON or CSV formats for training or plug in the model directly.

In conclusion, with MEGAnno, Meggie can explore her dataset using various heuristic-based or automated search functions and better understand the data corpus as she labels. She has the flexibility to iteratively update her schema as the project evolves. Using the widget, Meggie can finish the entire ML life cyle in the same Jupyter notebook.

4 Related Work

There exist numerous annotation tools that can support NLP tasks, which are extensively surveyed by Neves and Seva (2019, 2020). In this section, we focus on works that are closer to our flexible, exploratory, efficient, and seamless framework.

Flexible schema Unfortunately, most of existing tools are not designed for iterative schema development, and thus they are not flexible enough for evolving projects. Felix et al. (2018) and Kulesza et al. (2014) allow users to progressively define document classes by inspecting documents that are assigned to each class. But these works are more similar to interactive topic modeling or interactive classification, where users assign documents to classes, than document labeling, where users assign labels to documents. Our tool goes beyond document label refinement and supports a broader task of progressively defining annotation schema (e.g., additionally collecting span-level labels).

Exploratory labeling The concept of exploratory labeling is introduced by Felix et al. (2018) as using computational techniques to help users group documents into evolving labels. In our paper, we use the term "exploratory labeling" to refer to where exploratory data analysis and data annotation are iteratively conducted in the data understanding/exploration loop. Exploratory labeling can be beneficial because while labeling data, users gain insight into their dataset (Sun et al., 2017).

Efficient batch/bulk annotation A few tools offer a functionality to simultaneously assign labels to multiple spans within a record. For example, YEDDA (Yang et al., 2018) can annotate multiple span-level labels via command line. TALEN (Stephen Mayhew, 2018), a named entity tagging tool, has an entity propagation feature which annotates all mentions of an entity in a document at once. In contrast, users can annotate multiple records simultaneously using our Python API and a GUI widget.

Notebook widget Computational notebooks are frequently used by data analysts to iteratively write and edit code to understand data, test hypotheses, and build models (Head et al., 2019; Randles et al., 2017). Following the practice of mage (Kery et al., 2020) which extends Jupyter notebook with GUI widgets for specific tasks, our widget is designed to achieve flexible communication with the rest of ML development codes. Annotation tools which are implemented as Jupyter widgets include Pigeon (pig) and ipyannotate (ipy), but they only offer a simple label assignment feature.

5 Conclusion

In this paper, we present MEGAnno, an annotation framework designed for NLP researchers and practitioners. Through MEGAnno's programmatic interfaces and interactive widget, users can iteratively explore and search for interesting data subsets, annotate data, train models, evaluate and debug models within a Jupyter notebook without the overhead of context switch or data migration.

References

humanloop.com. https://humanloop.com/.

ipyannotate. https://github.com/ipyannotate/ipyannotate.

pigeon. https://github.com/agermanidis/pigeon.

Tivadar Danka and Peter Horvath. modAL: A modular active learning framework for Python. Available on arXiv at https://arxiv.org/abs/1805.00979.

Cristian Felix, Aritra Dasgupta, and Enrico Bertini. 2018. The exploratory labeling assistant: Mixed-initiative label curation with large document collections. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, UIST '18, pages 153–164, New York, NY, USA. Association for Computing Machinery.

Crowdflower Data for Everyone library. Twitter US Airline Sentiment(Version 4).

R. Stuart Geiger, Dominique Cope, Jamie Ip, Marsha Lotosh, Aayush Shah, Jenny Weng, and Rebekah Tang. 2021. "garbage in, garbage out" revisited: What do machine learning application papers report about human-labeled training data? *Quantitative Science Studies*, pages 1–33.

Andrew Head, Fred Hohman, Titus Barik, Steven M. Drucker, and Robert DeLine. 2019. *Managing Messes in Computational Notebooks*, pages 1–12. Association for Computing Machinery, New York, NY, USA.

Fred Hohman, Kanit Wongsuphasawat, Mary Beth Kery, and Kayur Patel. 2020. Understanding and visualizing data iteration in machine learning. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–13.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Mary Beth Kery, Donghao Ren, Fred Hohman, Dominik Moritz, Kanit Wongsuphasawat, and Kayur Patel. 2020. mage: Fluid moves between code and graphical work in computational notebooks.

Todd Kulesza, Saleema Amershi, Rich Caruana, Danyel Fisher, and Denis Charles. 2014. Structured labeling for facilitating concept evolution in machine learning. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 3075–3084, New York, NY, USA. Association for Computing Machinery.

Ines Montani and Matthew Honnibal. 2018. Prodigy: A new annotation tool for radically efficient machine teaching. *Artificial Intelligence*, to appear.

Mariana Neves and Jurica Seva. 2019. An extensive review of tools for manual annotation of documents. *Briefings in Bioinformatics*, 22(1):146–163.

Mariana Neves and Jurica Seva. 2020. Annotation-saurus: A searchable directory of annotation tools.

James Pustejovsky and Amber Stubbs. 2012. *Natural Language Annotation for Machine Learning: A guide to corpus-building for applications*. " O'Reilly Media, Inc.".

Sajjadur Rahman and Eser Kandogan. 2022. Characterizing practices, limitations, and opportunities related to text information extraction workflows: A human-in-the-loop perspective. CHI '22, New York, NY, USA. Association for Computing Machinery.

Bernadette M. Randles, Irene V. Pasquetto, Milena S. Golshan, and Christine L. Borgman. 2017. Using the jupyter notebook as a tool for open science: An empirical study. In *2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pages 1–2.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Dan Roth Stephen Mayhew. 2018. Talen: Tool for annotation of low-resource entities. In *ACL System Demonstrations*.

Yunjia Sun, Edward Lank, and Michael Terry. 2017. Label-and-learn: Visualizing the likelihood of machine learning classifier's success during data labeling. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces*, IUI '17, pages 523–534, New York, NY, USA. Association for Computing Machinery.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Jie Yang, Yue Zhang, Linwei Li, and Xingxuan Li. 2018. YEDDA: A lightweight collaborative text span annotation tool. In *Proceedings of ACL 2018, System Demonstrations*, pages 31–36, Melbourne, Australia. Association for Computational Linguistics.